PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06T	A2	(11) International Publication Number: WO 99/26198 (43) International Publication Date: 27 May 1999 (27.05.99)

(21) International Application Number: PCT/SG98/00092

(22) International Filing Date: 12 November 1998 (12.11.98)

(30) Priority Data:

60/065,805	14 November 1997 (14.11.97)	US
09/188,202	9 November 1998 (09.11.98)	US

(71) Applicant: NATIONAL UNIVERSITY OF SINGAPORE [SG/SG]; 10 Kent Ridge Crescent, Singapore 119260 (SG).

(72) Inventors: ONG, Kiem, Ching; Industry & Technology Relations Office (INTRO), National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260 (SG). TEH, Heng, Chuan; Industry & Technology Relations Office (INTRO), National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260 (SG). TAN, Tiow, Seng; Industry & Technology Relations Office (INTRO), National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260 (SG).

(74) Agent: APPLIED RESEARCH CORPORATION; Kent Ridge, P.O. Box 1016, Singapore 911101 (SG).

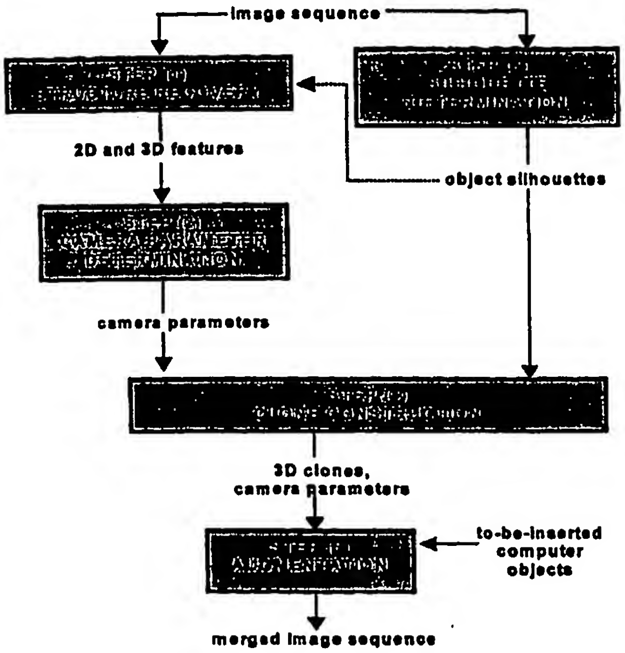
(81) Designated States: SG, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: SYSTEM AND METHOD FOR MERGING OBJECTS INTO AN IMAGE SEQUENCE WITHOUT PRIOR KNOWLEDGE OF THE SCENE IN THE IMAGE SEQUENCE

(57) Abstract

This invention describes a method for execution by a data processor that seamlessly merges objects into an image sequence to obtain a montage image sequence, without knowing *a priori* the content of the scene in the input image sequence. Such montage image sequence can be displayed to a viewer or recorded for future display. The method includes the steps of structure recovery, camera parameter derivation, silhouette computation, clone construction, and augmentation. Most of the steps are automated or semi–automated with minimal user intervention. The operation of the method includes the steps of interactively computing, displaying and recording merged image sequence in response to user commands to, for example, relocate inserted objects, change lighting parameters etc.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

SYSTEM AND METHOD FOR MERGING OBJECTS INTO AN IMAGE SEQUENCE WITHOUT PRIOR KNOWLEDGE OF THE SCENE IN THE IMAGE SEQUENCE

FIELD OF THE INVENTION

The present invention relates generally to computer graphics, image processing and augmented reality, and more particularly, to systems and methods for merging computer-generated objects or any images into a video or a sequence of photographs.

BACKGROUND OF THE INVENTION

The merging of computer-generated objects or other images into video sequences has important applications such as the blending of advertisement that exists in digital form into live broadcast images such that the advertisement is perceived to be present in the physical scene. Systems and methods that perform the merging automatically have been described in:

(1) U.S. Patent No. 5,436,672, "Video Processing System" issued to Gideon et al., 25 July 1995;

(2) U.S. Patent No. 5,488,675, "Stabilizing Estimate of Location of Target Region Inferred From Tracked Multiple Landmark Regions of a Video Image" issued to Hanna, 30 January 1996;

(3) U.S. Patent No. 5,491,517, "System for Implanting an Image into a Video Stream" issued to Kreitman et al., 13 February 1996;

(4) U.S. Patent No. 5,581,629, "Method for Estimating the Location of an Image Target Region from Tracked Multiple Image Landmark Regions" issued to Hanna et al., 3 December 1996;

(5) World Patent No. 97/00582, "System and Method of Real Time Insertions into Video Using Adaptive Occlusion with a Synthetic Reference Image" issued to Astle et al., 3 January 1997;

(6) World Patent No. 97/00581, "System and Method for Inserting Static and Dynamic Images into a Live Video Broadcast" issued to Di Cicco et al., 3

January 1997;

(7) World Patent No. 97/03517, "Methods and Apparatus for Producing Composite Video Images" issued to Beattie, 30 January 1997.

The entire contents of the above mentioned patents are fully incorporated herein by reference. These teachings assume prior knowledge of the video sequence such as the availability of landmarks, geometric and/or computer models of the scene. Such knowledge simplifies the task of occlusion resolution between objects in the video and objects/images merged into the video. Thus, they work in a specific environment such as a stadium, and are therefore not applicable to merging of objects/images in general, for example, in the case of home video sequence.

Other related teachings include: (1) World Patent No. 95/30312, "Improved Chromakeying System" issued to Tamir et al., 9 November 1995, and (2) World Patent No. 97/26758, "Method and Apparatus for Insertion of virtual Objects into a Video Sequence" issued to Goodman, 24 July 1997. They discuss the simple occlusion of inserting objects/images in front of all other objects in the video sequence. A qualitative approach on modeling the depth information of objects in a video sequence for the purpose of video compression is disclosed in World Patent No. 96/29678, "Method and Apparatus for Depth Modeling and providing Depth Information of Moving Objects" issued to Martens et al., 26 September 1996. Such qualitative assessment of depth is inadequate for our purposes.

There is currently little work that addresses the occlusion problem when scene model is not available. All the existing work found in the literature (such as (1) Breen, Whitaker, Rose and Tuceryan, *"Interactive occlusion and automatic object placement for augmented reality"*, Eurographics '96, 15(3), pp. C11—C22; (2) Wloka and Anderson, *"Resolving occlusion in augmented reality"*, Symposium on Interactive 3D Graphics Proceedings (New York), pp. 5—12, ACM Press, Apr 1995; and (3) Koch, *"Automatic reconstruction of buildings from stereoscopic image sequences"*, Eurographics '93, 12(3), pp. C339—C350, Oxford, UK, 1993) solve occlusion using dense depth maps derived from stereo image pairs when scene model is not available. This approach may be impractical and cumbersome at times because it requires two cameras with some fixed distance apart. The method proposed in this invention uses only monocular image sequence, which can be easily obtained using a video camera or a collection of photographs.

It is thus the object of this invention to provide methods and apparatus to seamlessly merge objects into an image sequence, without *a priori* knowledge about the real scene nor the availability of any scene model, and without much user intervention. The inserted objects may be static or dynamic like 2D windows or 3D virtual objects, and the image sequence may be monocular video sequence or a collection of photographs of a virtual or real environment that can be interpolated into a video sequence.

The invention can be used to build a software system to assist an architect evaluate the impact of his or her design by merging a computer model of the architecture into a video of the location where it is to be built. The augmented video conveys the idea more realistically compared to the conventional methods of using still images or physical models. Furthermore, changes in the design can be more readily realized and visualized in such a system. Besides architecture visualization, the invention can also find its usage in the movie industry, especially for special effects and in computer-animated films. Computer animation in real environment, which can be seen in today's TV commercials, can also be created using this invention.

SUMMARY OF THE INVENTION

The invention described herein satisfies the above-identified needs. It provides methods and apparatus to seamlessly merge objects into an image sequence to obtain a montage image sequence, without knowing *a priori* the scene content of the image sequence. Such montage image sequence can be shown to a viewer or recorded for future display. The inventors have determined that many details in the process of merging can be automated with little user intervention.

Occlusion is the phenomenon that allows one to only see the front object when two objects fall on the same line of sight. It is one of the most important visual cues that helps us perceive depth, thus survive in a 3D environment. In this invention, correct occlusion among the inserted objects and the objects in the image sequence is resolved to ensure convincing merging.

The basic principle employed in this invention to resolve occlusion is to construct a 3D clone for each important object of the scene in the image sequence. Such 3D clones, serving the same purpose of the real object in the image sequence, provide the proper occlusion to objects to be merged into the image sequence. Each 3D clone is usually an approximation of its corresponding real object since the complete information of the real object may not be available in the image sequence. More importantly, each 3D clone, when projected into the image sequence with the correct camera parameters, matches closely the image of the real object in the image sequence.

The process of constructing 3D clones is mostly automated with minimal user intervention. First, the method processes the image sequence to derive relative locations of important features (such as corners of buildings) as well as the camera parameters for each image in the image sequence. Second, the method obtains silhouettes of selected objects in selected images in the image sequence, either through automated or semi-automated segmentation techniques. With all this information, the recovered 3D features can be clustered according to the real objects they belong and then initial estimates of the 3D clones can then be derived, which are later improved using some optimization methods.

Specifically, there is disclosed a method for execution by a data processor that prepares an image sequence for merging seamlessly with other computer objects. The method includes the following steps:

(1) Structure Recovery—selected features of objects in the input image sequence are tracked whereby the spatial locality of these features are then computed.

(2) Camera Parameter Computation—from tracked features in the image sequence and their spatial locality information, the camera parameters for each image in the image sequence are computed.

(3) Silhouette Determination—the outline of the selected objects in the image sequence are computed automatically or semi-automatically.

(4) Clone Construction—3D clones of objects in the image sequence are computed using results from Step (1) to Step (3).

(5) Augmentation—with clones of objects in the image sequence, lighting parameters of the image sequence can be computed and other computer objects can be inserted into the image sequence.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 is a block diagram of an exemplary raster graphics systems;

FIG. 2 is a block diagram of the raster graphics system of FIG. 1 that shows the input data;

FIG. 3 is a flowchart that depicts the steps of the method of the invention;

FIG. 4 is a flowchart that depicts the operation of the method of the invention;

FIG. 5 is an approximate 3D clone that projects to real object silhouettes;

FIG. 6 is an initial estimate (γw_f) of camera position ($\boldsymbol{p}_f$);

FIG. 7 shows the clone construction process for an exemplary box in (a);

FIG. 8 is a mapping of viewing frustum to a 2D physical window;

FIG. 9 shows the slices of voxels used in the construction of clone;

FIG. 10 shows the rendering pipeline used to produce a merged image;

FIG. 11 is a transformation function that maps viewing coordinates to graphics subsystem coordinates;

FIG. 12 shows how the depth value is rectified when objects are rendered using different projection models.

DETAILED DESCRIPTION OF THE INVENTION

In our discussion that follows, frame and image are used interchangeably. **FIG. 1** illustrates an exemplary raster graphics system, which includes a main (Host) processor and a graphics subsystem. The Host processor executes an application program and dispatches graphics tasks to the graphics subsystem.

The graphics subsystem includes a pipeline of several components that perform operations necessary to prepare geometric entities for display on a raster display device. For the purposes of describing the invention, a model of the graphics subsystem is employed that contains the following functional units. It should be realized that this particular model is not to be construed in a limiting sense upon the practice of the invention.

A Geometric Processor unit performs geometric and perspective transformations, exact clipping on primitives against screen (window) boundaries, as well as lighting computations. The resulting graphics primitives, e.g. points, lines, triangles, etc., are described in screen space (integral) coordinates.

A Scan Conversion (Rasterization) unit breaks down the graphics primitives into raster information, i.e. a description of display screen pixels that are covered by the graphics primitives.

A Graphics Buffer unit receives, stores, and processes the pixels from the Scan Conversion unit in accordance with the method of the invention. The most common buffers associated to the Graphics Buffer unit are the frame buffer and the z-buffer. The frame buffer is where we store the RGB color components (in RGB mode) or the indices into a color map (in color map mode). These data decide the color of the pixels that will eventually be shown on the display unit. The z-buffer is used primarily for visible surface determination. It keeps track of the nearest z-coordinate (distance to the camera) at each pixel. Before we write any incoming pixel value into the frame buffer, we compare the z-coordinate of the geometry to that of the z-buffer. If the incoming z value shows that the new geometry is closer to the camera than the existing geometry, the value of the old pixel in the frame buffer and the old z value in the z-buffer are replaced by the new ones.

A Display unit receives pixels from the Graphics Buffer unit and transforms these pixels into information to either be displayed on the output device or recorded on some storage devices for future display.

Having described an exemplary graphics processing system that is suitable for use in practicing the invention, a description is now provided for a presently preferred method of merging objects with input image sequence.

Overview

The proposed method includes the following steps: (1) structure recovery, (2) camera parameter computation, (3) silhouette determination, (4) clone construction, and (5) augmentation. **FIG. 3** is a flowchart that depicts the flow of these steps of the invention.

We derive in Step (1) sparse 3D features from the scene using structure-from-motion algorithm from computer vision. First of all, some 2D image features in the starting frame are identified and subsequently tracked across the whole image sequence. These 2D image coordinates are then used to derive the 3D scene coordinates of the features. Following this, in Step (2) the recovered 3D features together with their corresponding 2D features are used to derive the unknown camera parameters like the zoom factor and the camera position in each frame. This is simply a minimization problem where the camera parameters are adjusted such that the difference between the tracked 2D features and the computed 2D projections of the corresponding 3D features is minimized. In Step (3), some frames are also selected, either automatically or semi-automatically, from the image sequence as key frames where the object silhouettes of some selected real objects are to be segmented. The output from Step (3) on object silhouettes may be utilized by Step (1), in which case Step (3) is run before Step (1) and (2), otherwise, Step (3) can run in parallel to Step (1) and (2). We then make use of these object silhouettes as well as the derived 3D feature points and camera parameters to construct in Step (4) approximate models, which are called ***clones*** to the real objects. Ideally, these clones, when projected onto each key frame of the image sequence, are to cover exactly the silhouettes of their respective real objects as shown in **FIG. 5**. With the clones, a common reference system between the real objects in the images and the inserted virtual objects can be established. The subsequent merging process in Step (5) is simply a depth comparison where proper occlusion is realized.

(1) Structure Recovery

There exist numerous methods in the literature that extract structure information from photographic or video images. One such method is from the motion of the camera or the objects in the scene. This method is well known as structure-from-motion problem in the computer vision community. Specifically, it designates the followings:

- Correspondence problem: establishing correspondence between the same features in consecutive frames.
- Reconstruction problem: 3D reconstruction of the scene and estimation of camera motion with the use of features, which have been successfully tracked.

Solving structure-from-motion problem using long image stream has two advantages. Firstly, feature correspondence problem between successive frames can be more easily solved due to small change in the image content. Secondly, data redundancy can be exploited to counteract noise in the images.

The structure-from-motion problem with perspective projection is inherently a non-linear minimization problem. This class of problems is susceptible to local minima in the parameter space and a good formulation is essential to better manage its complexity during minimization. In Szeliski and Kang, *"Recovering 3D shape and motion from image streams using non-linear least squares"* (Journal of Visual Communications and Image Representation, 5(1), pp. 10—28, Mar 1994) and Hu and Ahuja, *"Motion and structure estimation using long sequence motion models"*, (Image and Vision Computing, 11(9), pp. 549—570, Nov 1993), non-linear optimization techniques have been used to extract both 3D structure and camera motion from image streams. There are, however, researchers who relax the requirement of perspective projection to its approximations like orthography, scaled-orthography and paraperspective in order to convert the non-linear problem into a linear one. In particular, the factorization method proposed in

Tomasi and Kanade, *"Shape and motion from image streams under orthography: a factorization method"* (International Journal of Computer Vision, 9(2), pp. 137—154, Nov 1992), which assumes orthographic projection, has shown good results. In fact, Yu, Chen, Xu and Yachida, *"3D shape and motion by SVD under higher-order approximation of perspective projection"* (Proceedings of 13th International Conference on Pattern Recognition, pp. 456—460, Aug 1996) has managed to use Taylor series expansion to generalize the factorization method to higher-order approximations to perspective projection. All the above papers recover scene structure in terms of feature points. In Taylor and Kriegman, *"Structure and motion from line segments in multiple images"* (IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(11), Nov 1995), and Vieville and Faugeras, *"Feed-forward recovery of motion and structure from a sequence of 2D-line matches"* (IEEE International Conference on Computer Vision, pp. 517, Dec 1990), straight-line features are used to solve the structure-from-motion problem. The original factorization method has recently been extended to line features in Quan and Kanade, *"A factorization method for affine structure from line correspondences"* (Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 803—808, June 1996).

Factorization Method. The abovementioned methods on deriving structure and motion are applicable to our invention herein. As an example for subsequent discussion, we describe further the factorization method under orthography by Tomasi and Kanade (1992). Like other structure-from-motion algorithms that use point features, a set of 2D feature points are firstly selected in the first frame and then tracked across the entire image sequence. Assuming that P feature points have been successfully tracked across F frames, then a 2F×P measurement matrix of the image coordinates (u, v) can be formed. It is shown that, under orthography, the registered 2F×P matrix ($\tilde{\mathbf{W}}$) has rank 3 and can be decomposed into the camera orientation (**R**) and the 3D object shape (**S**).

$$\underset{2F\times P}{\tilde{\mathbf{W}}} = \underset{2F\times 3}{\mathbf{R}}\ \underset{3\times P}{\mathbf{S}}$$

This method is robust due to the use of singular value decomposition, which is numerically stable. Furthermore, shape information is derived without the use of intermediate camera-centered depth values, which are known to be noise-sensitive for distant objects. The output is the matrices R and S. R contains the relative orientation of the camera in each frame whereas S contains the recovered 3D feature point coordinates. The origin of the coordinate system is fixed at the centroid of the recovered 3D points and its orientation is arbitrarily aligned to the first frame's camera orientation.

Depth Reversal. The recovered 3D coordinates from the factorization method are only accurate up to a reflection about the Z = 0 plane. This implies that the depth values of the recovered 3D object shape are sometimes reversed, namely reflected about the Z = 0 plane. This occurs because there is inherently a sign ambiguity in the solution. Geometrically, the ambiguity arises because there is insufficient information to distinguish the front from the back of the object when only a set of points is tracked. To remedy this, users may need to examine the recovered 3D feature points and toggle the depth values when necessary.

Camera Motion. Even though the orthographic projection assumption is tolerable in some applications, such as architecture visualization where the camera that shoots the sequence is far away from the real scene, it does limit the kind of camera motion allowed. For example, the camera should not move towards or away from the scene in order to minimize perspective distortion. And, experiments have shown that camera should rotate around the target objects in the scene for at least 20 to 30 degrees for successful object shape recovery. On the other hand, when the camera is purely translated across the scene with its line of sight always in the same direction, the video sequence effectively only gives a 2D view of the objects and the factorization method thus cannot derive 3D structure of objects from such a sequence.

In another embodiment, the factorization method or any other structure-from-motion algorithms may be modified to work with video sequence that violates the specific requirements of the type of video sequence required by them. For example, a video sequence that moves towards or away from the scene can be corrected, with some user guidance, by incorporating scaling factors to correct each column in the matrix $\tilde{\mathbf{W}}$.

In yet another embodiment, this step of structure recovery can work with a collection of photographs instead of a video sequence. For a scene, given 2 or more photographs taken nearby each other, it is possible to derive other images between the two (see, for example, Mark, McMillan and Bishop, "*Post-rendering 3D Warping*", Proceedings of ACM Symposium on 3D Interactive Graphics, 1997, pp. 7—16). As such, a video sequence of the scene can be derived and treated as input to any structure-from-motion algorithms as before.

(2) Camera Parameter Computation

Structure-from-motion algorithms, in general, do not output a complete description of camera parameters. For example, the factorization method produces only the orientation **R** of the camera but not the position and zoom factor of the camera in each frame. In this case, given an image sequence of F frames, there are altogether 4F camera parameters to be derived, namely 3F camera position coordinates (3 coordinates per frame) and F zoom factors. Given the tracked 2D features and the recovered 3D features, we would like to compute the 4F parameters such that the 3D features are as closely projected to their respective 2D features as possible in all the frames. To achieve this, we can use the Levenberg-Marquardt minimization (see Press, Teukolsky, Vetterling and Flannery, *"Numerical Recipes in C: the art of scientific computing (2nd Edition)"*, Cambridge University Press, 1992) to minimize the error in distance between the tracked 2D features and the computed 2D projections (either orthographic or perspective) of the corresponding 3D features. For the case that the features are points, one possible error function is

$$\chi^2(\mathbf{a}) = \sum_{f=1}^{F}\sum_{p=1}^{P}\left\| \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix}_{\text{tracked}} - \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix}_{\text{computed}} \right\|$$

where

F : number of frames.

P : number of feature points.

a : vector containing the 4F camera parameters to be fitted.

$\begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix}_{\text{tracked}}$: tracked 2D feature point *p* in frame *f*.

$\begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix}_{\text{computed}}$: 2D projection of 3D feature point *p* in frame *f* given an estimate of **a**.

One other possible error function is to use maximum distance instead of summation in the above formula. For the case that the features are line segments, besides choosing appropriate points on these line segments to compute the error function based on distance as before, a different metric like area or angle between the tracked and the projected line features can also be employed.

The Levenberg-Marquardt minimization is an iterative method that requires an initial estimate of **a**. For the factorization method (refer to **FIG. 6**), the initial estimates of the camera positions (p_f) in each frame may be γw_f where γ is some constants (u_f, v_f and w_f define the camera orientation which is obtained from matrix **R** in the factorization method), and the initial estimates of zoom factors may be 1. One may terminate the iteration process when the fitted parameters do not improve a few times in a row.

In another embodiment, the 4F parameters may be reduced to 3F+1 if the camera is having the same zoom factor for all the frames.

(3) Silhouette Determination

In general, the structure-from-motion algorithm (Step (1)) only yields a sparse set of 3D features of the real objects, which is insufficient for us to construct 3D computer models, called clones. To this end, we further make use of the object silhouettes to help us construct the required 3D clones.

In one embodiment, a few frames are selected automatically or semi-automatically from the image sequence as key frames where the silhouettes of some selected real objects are to be segmented. As a guideline, a frame should be made a key frame when there is substantial geometrical change like the appearance and disappearance of facets on real objects in the image. Generally, the more key frames selected, the more accurate will be the constructed clones.

To date, a robust and fully automatic image segmentation algorithm that can correctly separate objects from the background is still elusive. Edge detection and region segmentation algorithms often fail when the intensity gradient at the object boundary is small. As a result, besides devising an automatic segmentation algorithm, researchers have focused on semi-automatic segmentation tools. These tools are often interactive and guided by users during the segmentation process. *Snakes* (Kass, Witikin and Terzopoulos, *"Snakes: active contour models"*, Proceedings of the First International Conference on Computer Vision, London, England, pp. 259—268, June 1987) and *Intelligent Scissors* (Mortensen and Barrett, *"Intelligent Scissors for image composition"*, Proceedings of SIGGRAPH '95, pp.191—198) are two such examples. On other occasions, it is desirable to track the segmentation boundary across multiple frames. Ueda, Mase and Suenaga, *"A contour tracking method using elastic contour model and energy minimization approach"* (IEICE Trans., vol. J75-D-II, No. 1, pp. 111—120, 1992), and Mitsunaga, Yokoyama and Totsuka, *"AutoKey: human assisted key extraction"* (Proceedings of SIGGRAPH '95, pp. 265—272) have attempted to tackle this problem with the use of *Snakes* and alpha values respectively. Nonetheless, these methods are not perfect at all times and user correction is usually needed when the segmentation boundary is wrongly tracked.

The method of this invention relies on correct object silhouette segmentation in the key frames for the construction of the clones. From the image sequence, users decide which of the objects need to build clones, and then obtain their silhouettes through automated or semi-automated tools such as those (or their variants) mentioned above.

In another embodiment, segmentation of objects in all frames may be used to build clones. This can be achieved as before by making all frames as key frames, or intelligently interpolate object silhouettes from a few key frames to all other frames by taking advantage of coherency in the image sequence. For example, object silhouette obtained in one frame may be used as the initial estimate of the silhouette of the same object in the next frame.

In yet another embodiment, segmentation results of this step may be utilized by structure recovery (Step (1)) to better select important features for tracking and to better track them across the image sequence.

(4) Clone Construction

The most important function that a clone to a real object serves is to occlude inserted objects. When an inserted object occludes a real object in the image, simple overlaying of the inserted object on top of the image will do the trick. This is not so straightforward when a real object in the image is to occlude the inserted object. We need a reference 3D clone of the real object in order to determine which part of the inserted object is to be occluded. Since the inserted object should only be occluded at the boundary of the real objects, the 3D clone should be constructed in such a way that its 2D projection onto the image plane exactly covers its object silhouette for every frame. The clone construction process can be broken down into 5 steps:

Step 4.1: Initial model definition

Suppose we want to construct a clone for the box in **FIG. 7a**. Based on the object silhouettes (see **FIG. 7b**) defined in the key frames, we first of all cluster the recovered 3D features from Step (1) to their respective real objects. We then define a rough model using the bounding box of the recovered 3D features. **FIG. 7c** shows the world-coordinate-axis-aligned bounding box of the derived 3D features (in this case, feature points) on the box. In another embodiment, an approximated bounding volume (rather than a bounding box) of the object may be supplied by users to expedite the subsequent processing.

Step 4.2: Model orientation determination

In some structure-from-motion algorithms such as the factorization method, the centroid-based coordinate system has an arbitrary orientation. Thus, the ground truth information is unknown, i.e., we cannot assume that the ground is on the Y = 0 plane and thus our initial rough model may not have the correct orientation. Having a correct orientation, though not critical, can speed up subsequent processing. As a result, users may be provided with an interactive mechanism to change the orientation of the initial model. **FIG. 7d** shows that user has interactively adjusted (select the square tag, near the upper right hand corner, and drag) the orientation of the bounding box so that it is closer to that of the white box.

Step 4.3: Model enlargement

Since a good clone is one that projects to its 2D object silhouette exactly, we need to ascertain that the initial model at least covers its object silhouettes in all the key frames, albeit over-cover. The following algorithm or variant of it (using other convenient bounding volume instead of bounding box) can systematically enlarge the model. **FIG. 7e** shows that the enlarged model projects to completely cover the object silhouette.

> Assuming that the initial model is bounded by bottom-left-far point (x_1, y_1, z_1) and top-right-near point (x_2, y_2, z_2) where $x_1 < x_2$, $y_1 < y_2$, and $z_1 < z_2$. The algorithm proceeds by going through all the key frames in turn to progressively enlarge the model. For each key frame, it computes the 2D (axis parallel) bounding box of the object silhouette and locates two points (possibly one point in degenerate cases), each on the object silhouette, that are furthest apart on each boundary edge of the bounding box. For each of these 8 (or less) points, it checks any enlargement is necessary as in the next paragraph.

For each of the 8 (or less) points, it casts a ray from the point on the image plane to the 3D space using the computed camera parameters from Step 2. If the ray intersects the current model, then no enlargement is necessary. Otherwise, the ray is outside the current model indicating that enlargement is necessary because the model does not project to cover this point. In this case, it finds the corner, say *C*, of the current model closest to the ray. Let 3D point (*x, y, z*) be the point on the ray closest to *C*. Then, the algorithm enlarges the current model by setting: $x_1 = \min(x_1, x)$; $y_1 = \min(y_1, y)$; $z_1 = \min(z_1, z)$; $x_2 = \max(x_2, x)$; $y_2 = \max(y_2, y)$; and $z_2 = \max(z_2, z)$.

Step 4.4: Model trimming

This step trims away 3D region on the model which does not project to the 2D object silhouette. To facilitate the trimming process, we use a voxel representation for the model. Each model is initially subdivided into voxels of similar size. We want to make each voxel, at the maximum, project to only cover a single pixel. This is because when a voxel projects to cover two or more pixels, there will be difficulty to decide whether to assign "IN" or "OUT" to the voxel when some of these pixels are inside the object silhouette while others are outside. Notice that the longest possible straight line enclosed within the 2D projection of a 3D voxel is the diagonal of its opposite corners. Suppose *x* is the dimension of the voxel, then the length of the diagonal of its opposite corners is $\sqrt{3}\,x$. To satisfy the one-pixel criterion, $\sqrt{3}\,x$ must be equivalent to one-pixel width.

The viewing frustum defined in the computer will be eventually mapped to the 2D physical window, which has the same dimension as the real image. Thus, referring to **FIG. 8**, we have

$$\sqrt{3}x = \frac{\text{frustum height}}{\text{image height}}$$

$$\Rightarrow \quad x = \frac{\text{frustum height}}{\sqrt{3} \times \text{image height}}$$

All the voxels of the model are initially assigned "OUT" to indicate that they are not part of the model. We then employ a ray-casting algorithm that casts ray from the image plane to the 3D space to trim the model. The word "trim" actually means form the model rather than trim away unwanted region from the model in the following algorithm.

```
for each pixel on the boundary of object silhouette
    cast a ray from pixel and intersect ray with model;
    for each voxel along the ray's path
            set voxel property to "IN";  /* boundary voxels */
find the closest face of the bounding box to the camera and examine the
    voxels in slices (refer to FIG. 9);
for each voxel enclosed by the boundary voxels
    set voxel property to "IN";
```

The above is repeated for each of the key frames where object silhouettes have been defined. Another way to look at the whole trimming process is by solid object intersection. We first of all define a valid 3D region (model defined after Step 4.3) where intersection is allowed. The resultant clone is essentially the intersection of the object-frustum back-projected from the object silhouettes in the key frames to the 3D space. **FIG. 7f** shows the constructed clone, which is represented using voxels. Notice that the model has missed out boundaries at top and bottom of the box.

Step 4.5: Dynamic patch voxel derivation

Up to this point, we have obtained 3D clones, which are trimmed using their respective object silhouettes in the key frames. Due to inaccuracy in the computed camera parameters (camera position, orientation and zoom factor), each such clone may be over-trimmed at certain region, especially near the boundary. This implies that the clone will no longer project to cover exactly the object silhouette in some key frames and visual error may occur at the occluding boundary between real and inserted objects in the merged image sequence. To remedy this, we make use of a set of dynamic 3D patch voxels, which have the same voxel dimension as the clone. For each key frame, we locate the missing region on the clone, which will otherwise make the clone project perfectly to the object silhouette. We later encode this missing region using the patch voxels.

The challenge to this problem is, given that a clone does not project exactly onto the silhouette, how to reinstate the missing 3D region on the clone. We know that the missing region is still within the bounding box of the clone and must be geometrically close to the over-trimmed model. As such, the following algorithm can be used to extrapolate the model.

> The patch voxels needed for each key frame is computed with two passes through all pixels within the object silhouette.
>
> Let's discuss the first pass. For each pixel, it casts a ray from the pixel on image plane to the 3D space (using computed camera parameters). If the ray does not intersect the clone, then over-trimming of the clone has occurred for this pixel and the pixel is marked. Otherwise, if the pixel has any of its 8 neighboring pixels already marked, it computes d_1 and d_2, the nearest and furthest *z*-coordinates, respectively, of the clone intersected by the ray. Since the status of the 8 neighboring pixels may not be available when a pixel is examined, some backtracking algorithms are required to backtrack to the pixel should any of its neighboring pixels is subsequently marked.
>
> In the second pass, we create patch voxels for pixels marked in the first pass. For each such pixel, it estimates the d_1 (and d_2 respectively) of the pixel as the average of the d_1 values (and d_2 values respectively) of surrounding 8 (or less) pixels. Note that the estimation has to be done in some order of pixels closest to pixels with known d_1 and d_2 values and so on. Next, it casts a ray from the pixel so as to encode the ray segment between d_1 and d_2 as patch voxels.

In practice, the last statement on casting a ray is replaced by casting more than one ray per pixel. The concept is similar to super-sampling in ray-tracing. This is to ensure that any potential patch voxel, which may belong to the missing region is not overlooked.

Now, each key frame has a set of 3D patch voxels, which can be employed in the following manner. During merging of objects into image sequence, when the frame is a key frame, we use the over-trimmed model together with the key frame's patch voxels as clone to its corresponding object. This ensures that the object silhouette is exactly covered by the clone. In one embodiment, for intermediate frames, which are not key frames, we may define the clone to be the over-trimmed model together with the patch voxels of the *nearest* key frame. So, when we step through the image sequence, the clone, which is the over-trimmed model, is dynamically patched up using patch voxels belonging to different key frames. **FIG. 7g** and **FIG. 7h** show two different views of the clone with patch voxels drawn. In another embodiment, the content of the non-key frames may be analyzed automatically to include only selected patch voxels, possibly from all the key frames.

The over-trimming of a clone is the result of either occlusion of the object in some key frame or inaccuracy in the computed camera parameters. In yet another embodiment, the accuracy of the computed camera parameters can be improved by formulating another optimization process to minimize, using the Levenberg-Marquardt algorithm, the number of patch voxels due to inaccuracy in the computed camera parameters.

In another embodiment, besides using 3D voxels to represent a clone, another more time-efficient method is to simply use a 3D plane of any polygonal shape. To serve as clone, this 3D plane will be masked by the object silhouettes in the frames, like being cut using cookie cutters. This method requires the object silhouettes in the key frames to be interpolated to all the non-key frames. For a real object, its silhouettes in the key frames are firstly broken into vertices and the vertices from the silhouettes of two consecutive key frames are paired up according to proximity. (A vertex can be defined as a pixel location where the boundary changes direction.) This effectively generates a 2D morphing path for the vertices on the silhouette boundary. These vertices, when translate to their respective matched counterparts from one key frame to the next across the entire video sequence, yields the interpolated silhouettes in the non-key frames. In short, given the object silhouettes in the key frames, they will be interpolated in the non-key frames through 2D vertex morphing. With the availability of the object silhouettes in all the frames (both key and non-key frames), the 3D plane is masked by the silhouettes so that only portion that falls within the silhouette is used as clone. The position of the 3D plane is determined as follows. Using a segmented silhouette from any of the key frame, 3D feature points which project to the silhouette region are used to compute a 3D object centroid. The 3D plane will be centered at this centroid with its orientation always directly facing the camera. The size of the 3D plane is computed in such a way that when the plane is projected to the screen, it at least covers the entire silhouette in the frame. With such an approach, the 3D clone (in terms of a 3D plane masked by the silhouettes) will project to its object silhouette in every frame.

(5) Augmentation

Lighting parameters of the image sequence are important in rendering computer objects so that these objects blend nicely to the scene in the image sequence. To obtain the lighting parameters, we can use the constructed 3D clones as a basis together with a variant of the approach of Schoeneman, Dorsey, Smits, Arvo and Greenberg, "*Painting with light*" (Proceedings of SIGGRAPH '93, pp. 143—146) that computes the lighting parameters of a given targeted illumination of a scene. Other relevant references in the area also include: (1) Fournier, Gunawan and Romanzin, "*Common illumination between real and computer generated scenes*", Proceedings of Graphics Interface, 1993, pp. 254—261; (2) Nakamae, Harada, Ishizaki and Nishita, "*A montage method: the overlaying of the computer generated images onto a background photograph*", Computer Graphics, vol. 20, 1986, pp. 207—214, (3) Poulin and Fournier, "*Lights from highlights and shadows*", Proceedings of ACM Symposium on 3D Interactive Graphics, 1992, 31—38, and (4) Chevrier, "Handling interactions between real and virtual worlds", Proceedings of Computer Graphics International, 1996, pp. 74—83. Alternatively, in most cases of outdoor scenery, the lighting parameters can be approximated with a single directional light source.

With the above, each image of the merged image sequence can be produced with the following rendering pipeline.

Step 5.1: Display image in frame buffer (refer to **FIG. 10a**).

Step 5.2: Render each clone (both over-trimmed model and patch voxels) in only the z-buffer. This step ensures that the real object has its depth information registered in the Graphics Buffer unit (refer to **FIG. 10b**).

Step 5.3: Render each inserted object in both frame buffer and z-buffer. Each of these computer objects will appear occluded in the merged image if it is behind some real objects (refer to **FIG. 10c**).

In order to ensure that the clones still conform to the way they are built, we should use the same projection model (either orthographic or perspective) that constructed the clone to render the clones to the z-buffer in Step 5.2. The computer objects can be rendered using either orthographic or perspective projection.

In different projection models, the z-values used in depth comparison may no longer conform to the correct viewing order. For example, assuming that point A and B fall on the same line of sight and A is closer to the camera. When both of the points are rendered with the same projection model, point A will always occlude point B. However, when A is rendered with perspective and B with orthographic projection, the visibility order may be wrong even though point A is still closer to the camera. This is because the mapping of the depth values from the viewing coordinate to the graphics subsystem coordinate is different for orthographic and perspective. Under orthography, $z_{system} = \frac{-2z_{view} - (\text{far} + \text{near})}{\text{far} - \text{near}}$, whereas under perspective, $z_{system} = \frac{far + near}{far - near} + \frac{2\,far\,near}{z_{view}(far - near)}$, where

z_{system} : z value in graphics subsystem coordinate;
z_{view} : z value in viewing coordinate;
near : distance between camera and near plane of viewing frustum;
far : distance between camera and far plane of viewing frustum.

If we consider only the values within the viewing frustum, then ($-far \le z_{view} \le -near$) and ($-1 \le z_{system} \le 1$). As shown in **FIG. 11**, the mapping is linear under orthographic projection but non-linear under perspective projection. The scenario given above about A and B occurs when they take the values indicated in the graph: A is in front of B when they are both rendered with the same projection model, but A appears behind B when A is rendered with perspective and B with orthographic projection.

If we render clones using orthographic and inserted objects using perspective projection, without any rectification, the inserted objects may well be occluded by the clones even though they are placed nearer to the camera than the clones. In order to restore the correct depth order, the depth values of the clones can be adjusted in the following manner. Suppose that the clone's depth values spans R_1 in the viewing coordinate. With perspective projection, R_1 is mapped to T in the graphics subsystem coordinate. Since we use orthographic projection to render the clone, we therefore must compute the range of values in the viewing coordinate that maps to T under orthography. Refer to **FIG. 12**, R_2 turns out to be the required range. We then translate and scale the depth values of the clone from R_1 to R_2 so that when the inserted objects are rendered with perspective projection, the correct depth order between the clone and inserted objects is preserved.

In the above discussion, the constructed clones are assumed to be strictly opaque. In another embodiment, we can employ the technique of alpha blending to realize the translucent property of some real world objects like glass.

OPERATION OF THE METHOD OF INVENTION

FIG. 4 is a flowchart that depicts the operation of the method of the invention. First, the system loads an image sequence. This may be an image sequence of a site where a new infrastructure is to be constructed. The input image sequence may be processed by image processing tools that, for example, filter or mask out unnecessary details or noise before further processing by the method described in this invention.

For each of the object of importance in the image sequence, a clone is constructed as depicted in **FIG. 3**. Some user interactions may be necessary in the silhouette computation, and further information like the speed of the camera or the path of the camera while taking the image sequence is helpful to ease the computation of structure and camera parameters recovery. In some cases, a portion of the image sequence is sufficient to complete the structure and motion recovery, and the computed result may be extended to the full image sequence. The image sequence and clones for objects are stored in the memory as shown in **FIG. 2**. At this point, the system is ready to begin an interactive display session with a user.

Commands from the user may include the change of the position and the lighting parameters for the inserted object as well as the viewing of the image sequence from a novel viewpoint (see, for example, Horry, Anjyo and Arai, "*Tour into the pictures: Using a spidery mesh interface to make animation from a single image*", Proceedings of SIGGRAPH '97, pp. 225—232). In response to a command, the system accesses the memory to obtain clone and image information and computes, using interpolation when necessary, the merged image sequence which is either displayed by the graphics subsystem on the display unit or recorded for future display. In some cases, collision detection between the inserted objects and the clones in the image sequence may also be computed. This is useful when an inserted object is to sit on the real floor/ground in the image sequence for example.

CLAIMS

Thus, while the invention has been particularly shown and described with respect to preferred embodiments thereof, it will be understood by those skilled in the art that changes in form and details may be made therein without departing from the scope and spirit of the invention.

Having thus described our invention, what we claim as new, and desire to secure by Letters Patent are:

1. A method for execution by a data processor to merge objects into an image sequence, without *a priori* knowledge about the scene nor the availability of any scene model and without much user intervention, comprising the steps of:

   processing the input image sequence, which is either a monocular video sequence or a collection of photographs of a virtual or real environment that can be interpolated into a video sequence, to produce 3D clones of objects in the image sequence;

   merging objects stored in the data processor into the input image sequence to produce the merged image sequence, as if the objects were originally existed at some specified locations in the scene captured by the image sequence, to be displayed to a viewer or recorded for future display.

2. A method as set forth in claim 1 wherein the step of processing the input image sequence comprises the steps of:

   structure recovery in which selected features of objects in the input image sequence are tracked so as to compute the spatial locality of these features;

   camera parameter computation in which tracked features and their computed spatial locality are used to derive the camera parameters of each image in the image sequence;

   silhouette determination in which the outline of selected objects in the image sequence are computed automatically or semi-automatically;

   clone construction in which 3D clones of objects in the image sequence are computed using results from the previous three steps;

augmentation in which lighting parameters of the image sequence are derived and merged image sequence is produced.

3. A method as set forth in claim 2 wherein the step of structure recovery includes a step of using structure-from-motion algorithm from computer vision to track important features in the image sequence from frame to frame, and to derive 3D spatial locations of these features and possibly some camera parameter information for each frame.

4. A method as set forth in claim 3 wherein the step of structure-from-motion algorithm includes a step to modify, automatically or semi-automatically, the data of the tracked features in the image sequence to conform to the specific requirements of the type of image sequence acceptable by the said structure-from-motion algorithm.

5. A method as set forth in claim 3 further including a step of preprocessing input image sequence to filter or mask out unnecessary details or noise before processing by the structure-from-motion algorithm.

6. A method as set forth in claim 3 further including a step of using techniques of interpolation to derive, from photographs taken nearby each other of a scene, video sequence of the scene for use in structure-from-motion algorithm.

7. A method as set forth in claim 2 wherein the step of structure recovery includes a step of reflecting the recovered 3D features about the Z=0 plane.

8. A method as set forth in claim 2 wherein the step of camera parameter computation includes a step using The Levenberg-Marquardt algorithm to minimize an error function between the tracked 2D features and the projected 2D features of the corresponding computed 3D features.

9. A method as set forth in claim 8 wherein the error function, in the case of tracked feature points, is either the sum or the maximum distance, and, in the case of tracked feature edges, is either the sum or the maximum distance

between some selected points on feature edges, or area or angle formed between tracked and computed 2D feature edges.

10. A method as set forth in claim 8 wherein the Levenberg-Marquardt algorithm includes the use of those computed camera parameters from the structure-from-motion algorithm as the initial estimates of the camera parameters.

11. A method as set forth in claim 8 further including a step to incorporating user input in simplifying the minimization formulation.

12. A method as set forth in claim 2 wherein the step of silhouette determination includes a step of key frame selection and a step of image segmentation in each key frame.

13. A method as set forth in claim 12 wherein the step of key frame selection includes a step to select a frame as a key frame if there is substantial geometrical change, such as the appearance and disappearance of facets on real objects in the image, between this frame and the previous frame in the image sequence.

14. A method as set forth in claim 12 wherein the step of key frame selection includes a step to select the first and the last frame in the image sequence as key frames.

15. A method as set forth in claim 12 wherein the step of image segmentation includes a step of using automatic and semi-automatic image segmentation algorithm in image processing to separate objects from the background in key frames.

16. A method as set forth in claim 15 further including a step to use object silhouettes from key frames to automatically segment objects in other frames by taking advantage of coherency in the image sequence.

17. A method as set forth in claim 3 wherein the step of tracking 2D features includes a step of using object silhouettes of claim 15 and claim 16 in selecting

important 2D features and in assisting their tracking across the image sequence.

18. A method as set forth in claim 2 wherein the step of clone construction for an object in the scene comprises the steps of:

defining initial clone using the recovered 3D features;

orientating initial clone to better align with the real object it represents;

enlarging initial clone to at least covers its object silhouettes in all key frames;

trimming 3D regions of the initial clone which does not project to the object silhouette in each key frame;

deriving dynamic patch voxels of the 3D clones for regions that are over-trimmed in the previous step.

19. A method as set forth in claim 18 wherein the step of defining the initial clone includes a step of first clustering the recovered 3D features in claim 3 to their respective real objects, and then defining a rough model using the bounding box or other convenient bounding volume of the recovered 3D features.

20. A method as set forth in claim 18 wherein the step of orientating initial clone includes a step of interactive mechanism to change the orientation of the clone.

21. A method as set forth in claim 18 wherein the step of enlarging initial clone of an object comprises steps:

to examine in turn the silhouette of the object in each key frame,

to cast rays, using the computed camera parameters, from extreme boundary points on the silhouette to the 3D space, and

to enlarge the clone accordingly when the ray misses the current clone.

22. A method as set forth in claim 18 wherein the step of trimming the clone of an object includes a step of computing the representation of the clone as voxels.

23. A method as set forth in claim 22 wherein the dimension of the voxel is determined so that each voxel, at the maximum, projects to only cover a single

pixel on the display unit.

24. A method as set forth in claim 18 wherein the step of trimming the clone of an object comprises steps:

to examine in turn the silhouette of the object in each key frame,

to cast one or more rays, using the computed camera parameters, from each pixel on the boundary of object silhouette to the 3D space, and

to trim away the region of the clone which does not project to the object silhouette.

25. A method as set forth in claim 18 wherein the step of deriving dynamic patch voxels of a clone for a key frame comprises steps:

to cast one or more rays to the 3D space, using the computed camera parameters, from each pixel within the object silhouette, and

to generate the patch voxels for each pixel whose ray cast misses the clone, using the known depth information of neighboring pixels.

26. A method as set forth in claim 18 whereby the dynamic patch voxels of key frames are utilized to patch up the clone for the nearby non-key frames.

27. A method as set forth in claim 18 further including a step to improve the accuracy of the computed camera parameters by minimizing the number of patch voxels using The Levenberg-Marquardt algorithm.

28. A method as set forth in claim 2 wherein for a more time-efficient clone construction process, a simple 3D plane is used which will be masked by the object silhouettes of a particular real object in the scene, so that only relevant portion of the 3D plane which falls within the silhouette is used as clone.

29. A method as set forth in claim 28 whereby the object silhouettes in the key frames are interpolated to the non-key frames using 2D vertex translation (morphing) and the 3D plane is positioned at the centroid of the 3D feature points that project to the silhouette of a particular key frame and is always oriented to be directly facing the camera.

30. A method as set forth in claim 2 wherein the step of augmentation to each image in the image sequence comprises the steps of:

obtaining lighting parameters of the image sequence,

displaying image in frame buffer,

rendering each clone (over-trimmed model and patch voxels) in only the z-buffer, and

rendering each computer object to be inserted in both frame buffer and z-buffer.

31. A method as set forth in claim 30 wherein the step of rendering clone and inserted objects includes a step to rectify the depth values of the clone and inserted objects when different projection models like orthographic and perspective are used in the rendering.

32. A method as set forth in claim 30 wherein the step of rendering clone includes a step to render translucent or strictly opaque clone, using the technique of alpha blending.

33. A method as set forth in claim 1 further including a step to obtain a new image at a novel viewpoint, from existing images in the image sequence, using interpolation techniques.

34. A method as set forth in claim 1 wherein the objects stored in the data processor to be inserted into the input image sequence is either static or dynamic (changes shape from frame to frame) like 2D windows or 3D computer objects.

35. Graphics display apparatus for displaying and recording merged image sequence, comprising:

input means for capturing using, for example, a video camera an input image sequence, and creating using, for example, a modeling tool computer objects to be inserted into the image sequence;

memory means for storing image sequence, objects to be inserted into the image sequence, and merged image sequence;

means, responsive to a command from a user of the apparatus, for indicating the change of the position of the inserted object, lighting parameters for the merged image sequence, viewing of the merged image sequence from a novel viewpoint, etc.;

processing means, coupled to and responsive to the indicating means for accessing the memory means to display or record merged image sequence.

36. Graphics display apparatus as set forth in claim 35 wherein said processing means includes means for processing the image sequence, as set forth in claim 2 so as to produce the merged image sequence.

37. Graphics display apparatus as set forth in claim 36 further including means for storing array data structures containing 2D features, 3D features, object silhouettes, and camera parameters for each image in the image sequence.

38. Apparatus for merging into an image sequence, without *a priori* knowledge about the real scene nor the availability of any scene model, and without much user intervention, comprising:

means for processing the input image sequence, which is either a monocular video sequence or a collection of photographs of a virtual or real environment that can be interpolated into a video sequence, to produce 3D clones of objects in the image sequence;

means for merging objects stored in the data processor into the input image sequence to produce the merged image sequence, as if the objects were originally existed at some specified locations in the scene captured by the image sequence, to be displayed to a viewer or recorded for future display.

FIG. 1

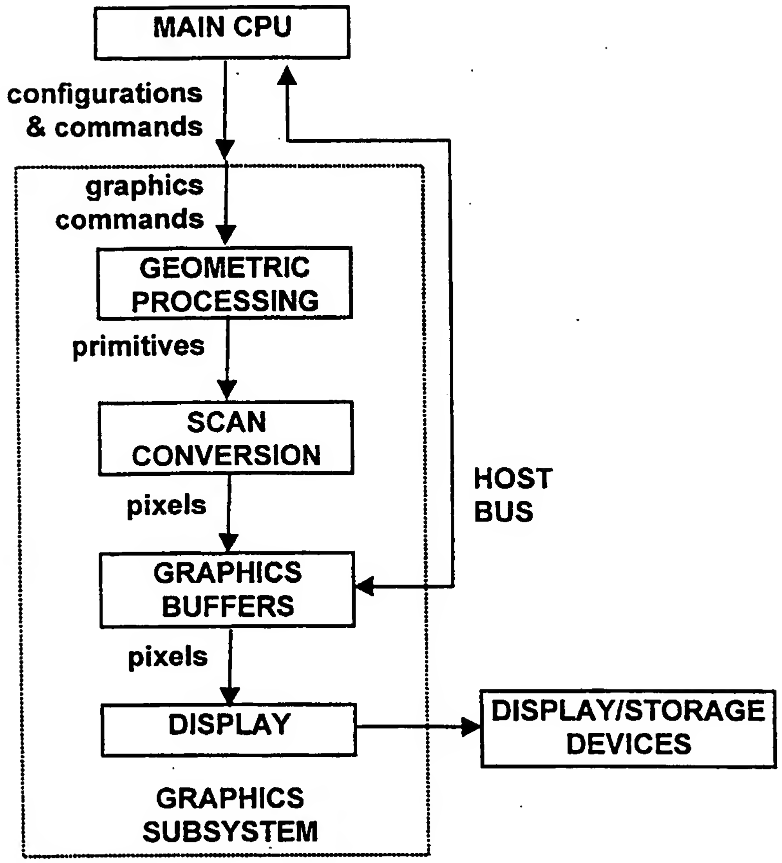

MAIN CPU
configurations & commands
graphics commands
GEOMETRIC PROCESSING
primitives
SCAN CONVERSION
pixels
GRAPHICS BUFFERS
pixels
DISPLAY
GRAPHICS SUBSYSTEM
HOST BUS
DISPLAY/STORAGE DEVICES

FIG. 2

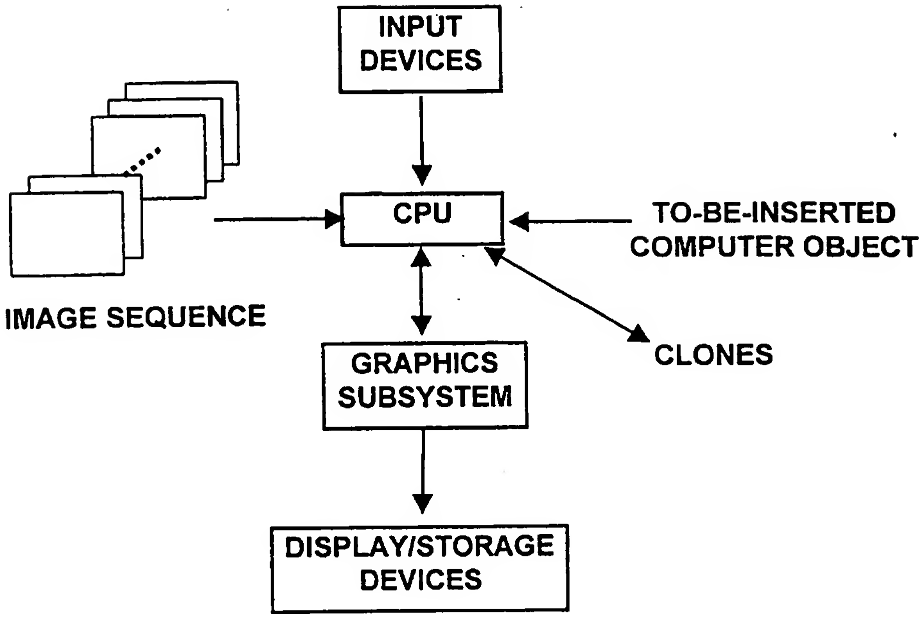

INPUT DEVICES
CPU
TO-BE-INSERTED COMPUTER OBJECT
IMAGE SEQUENCE
CLONES
GRAPHICS SUBSYSTEM
DISPLAY/STORAGE DEVICES

FIG. 3

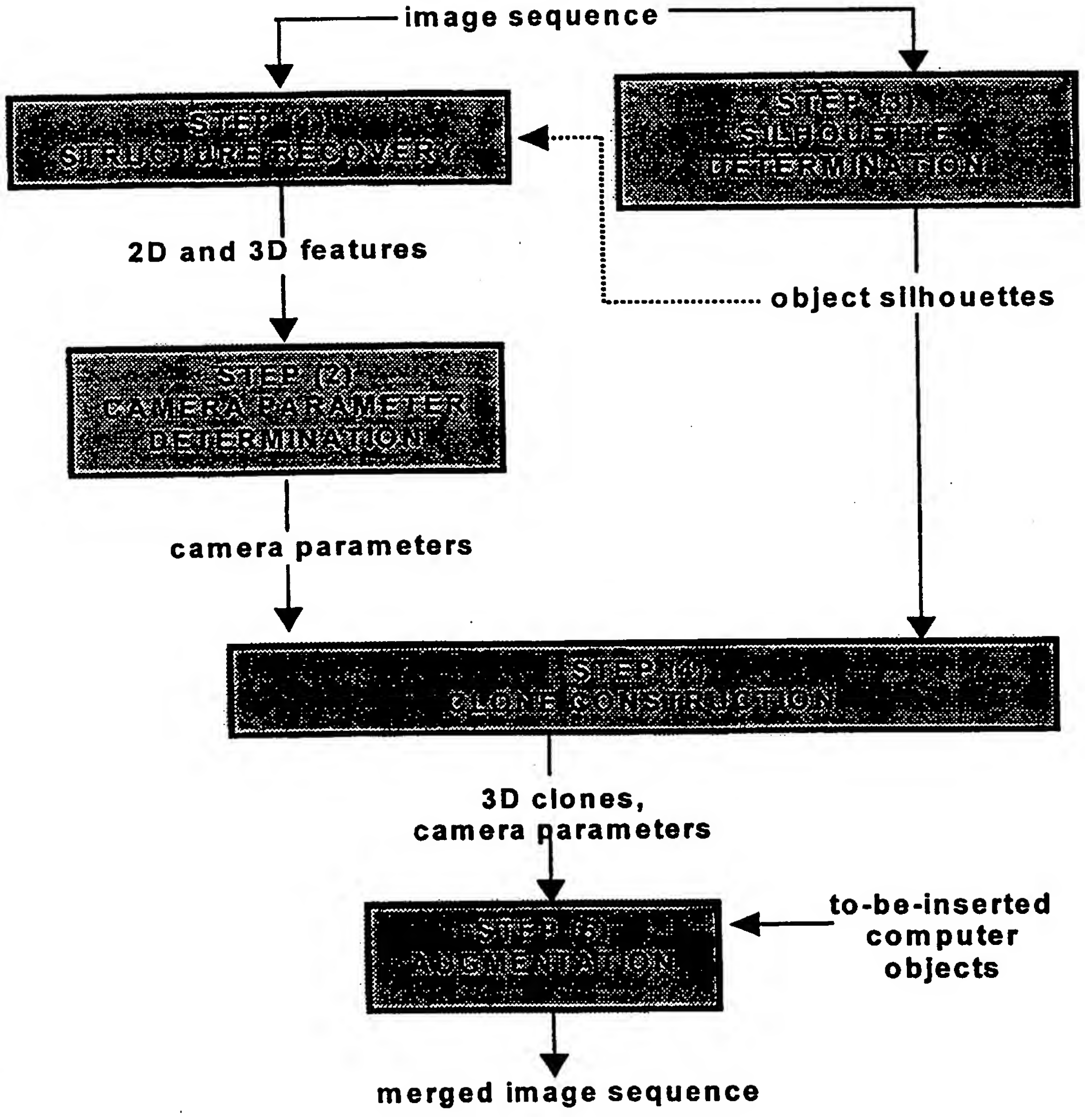


image sequence
STEP (1)
STRUCTURE RECOVERY
STEP (3)
SILHOUETTE
DETERMINATION
2D and 3D features
object silhouettes
STEP (2)
CAMERA PARAMETER
DETERMINATION
camera parameters
STEP (4)
CLONE CONSTRUCTION
3D clones,
camera parameters
STEP (5)
AUGMENTATION
to-be-inserted
computer
objects
merged image sequence

FIG. 4

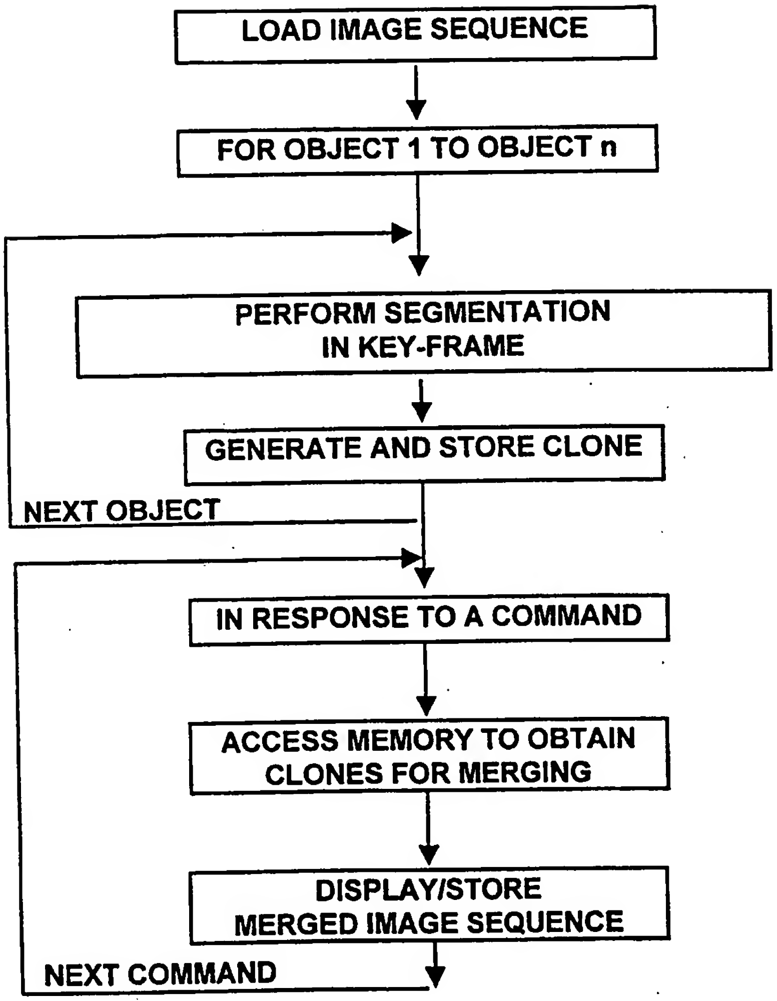

LOAD IMAGE SEQUENCE
FOR OBJECT 1 TO OBJECT n
PERFORM SEGMENTATION IN KEY-FRAME
GENERATE AND STORE CLONE
NEXT OBJECT
IN RESPONSE TO A COMMAND
ACCESS MEMORY TO OBTAIN CLONES FOR MERGING
DISPLAY/STORE MERGED IMAGE SEQUENCE
NEXT COMMAND

FIG. 5

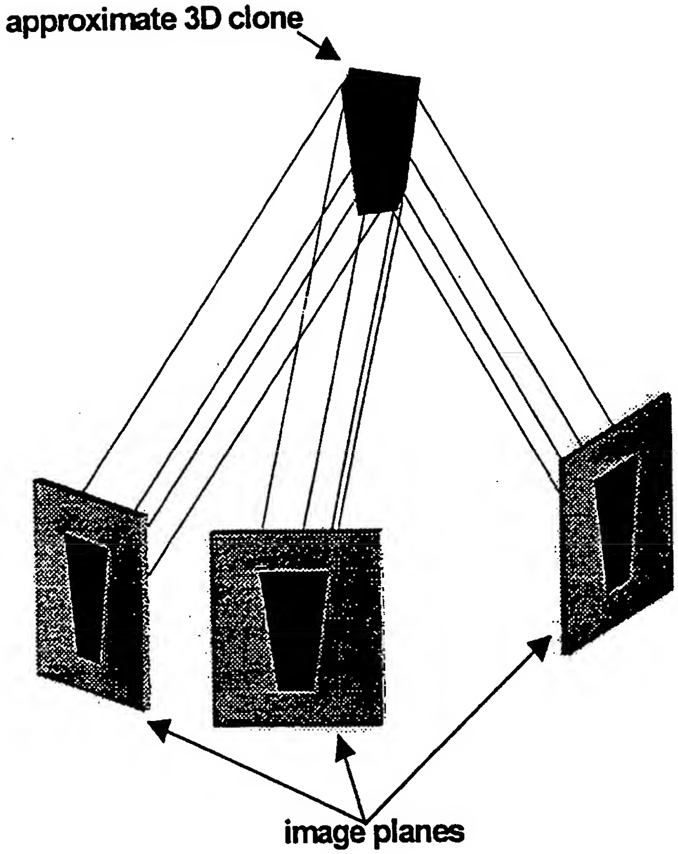

approximate 3D clone
image planes

FIG. 6

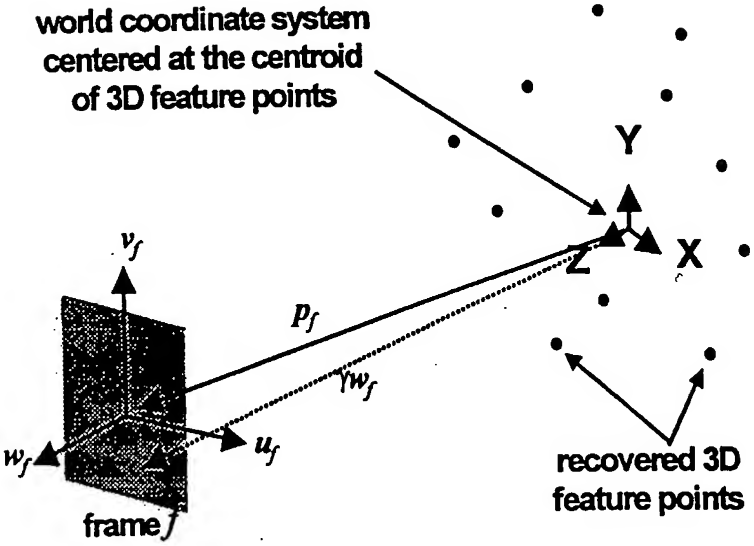

world coordinate system
centered at the centroid
of 3D feature points
Y
X
Z
v_f
p_f
γw_f
w_f
u_f
frame f
recovered 3D
feature points

FIG. 7

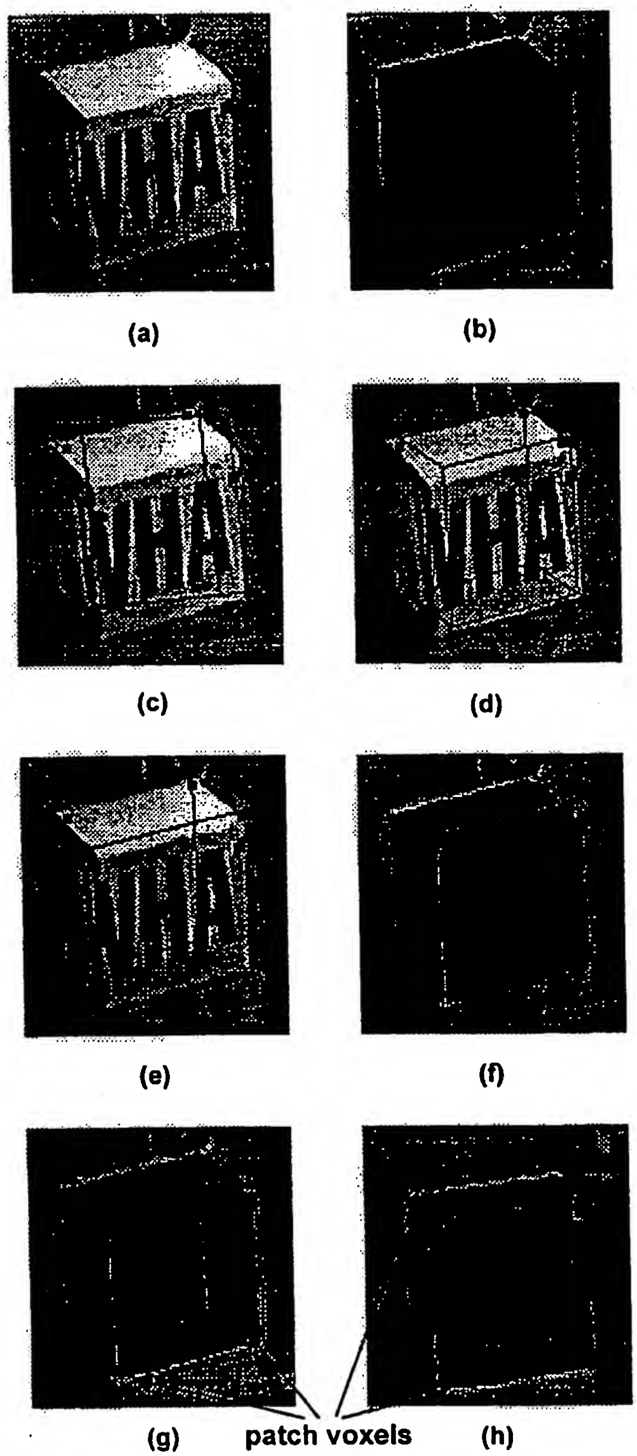


FIG. 8

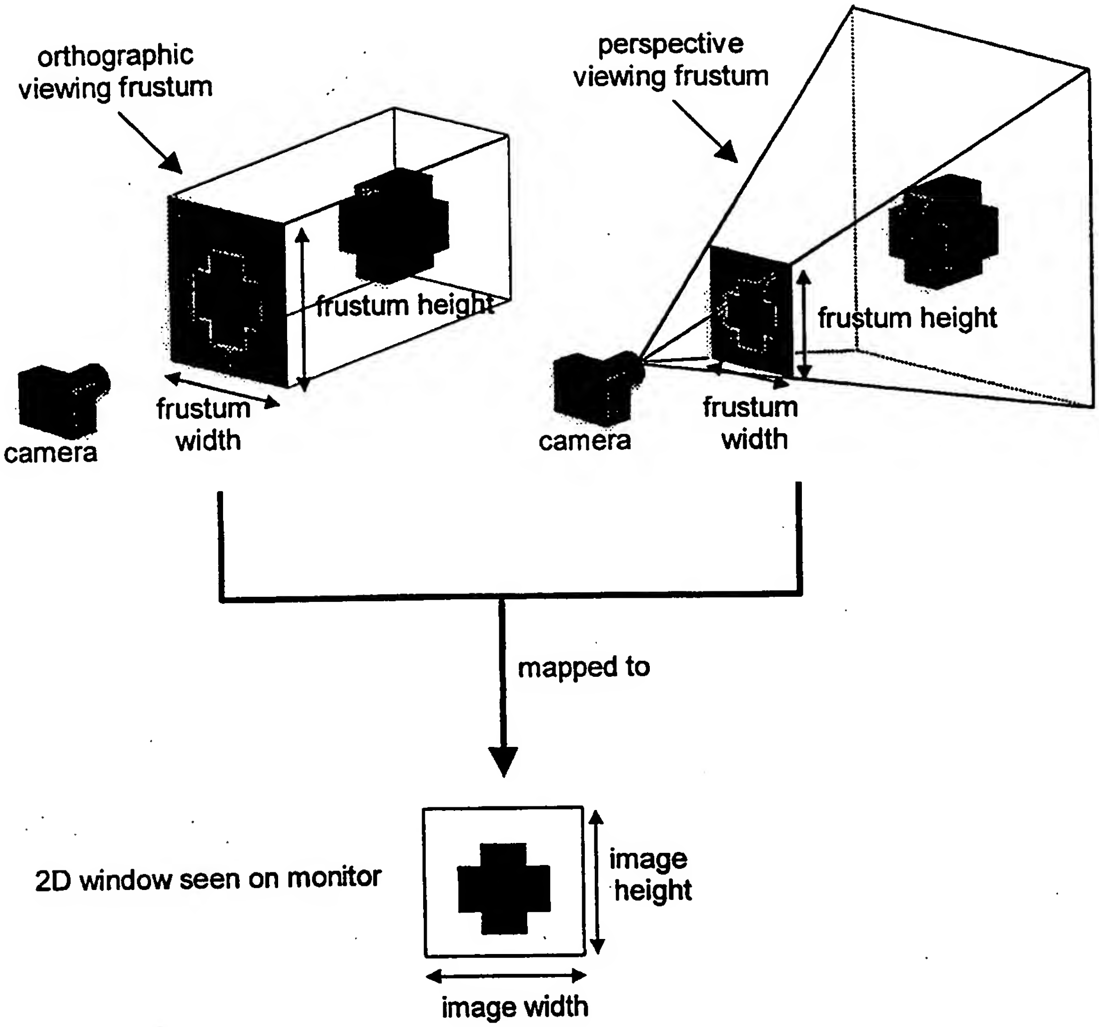


image width = image height × aspect ratio
frustum width = frustum height × aspect ratio

FIG. 9

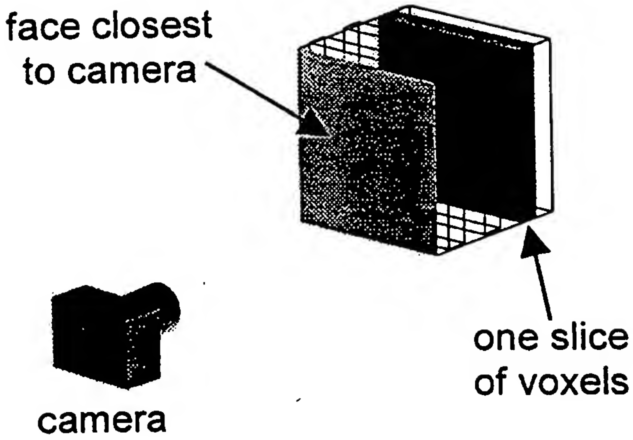
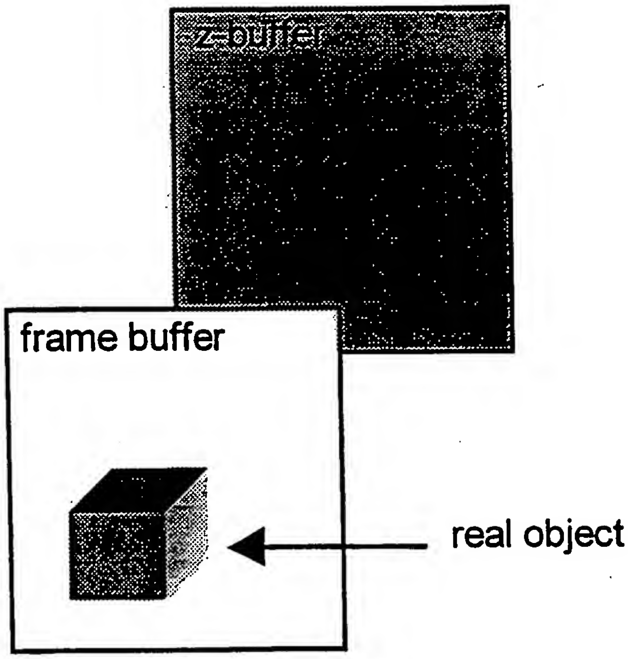
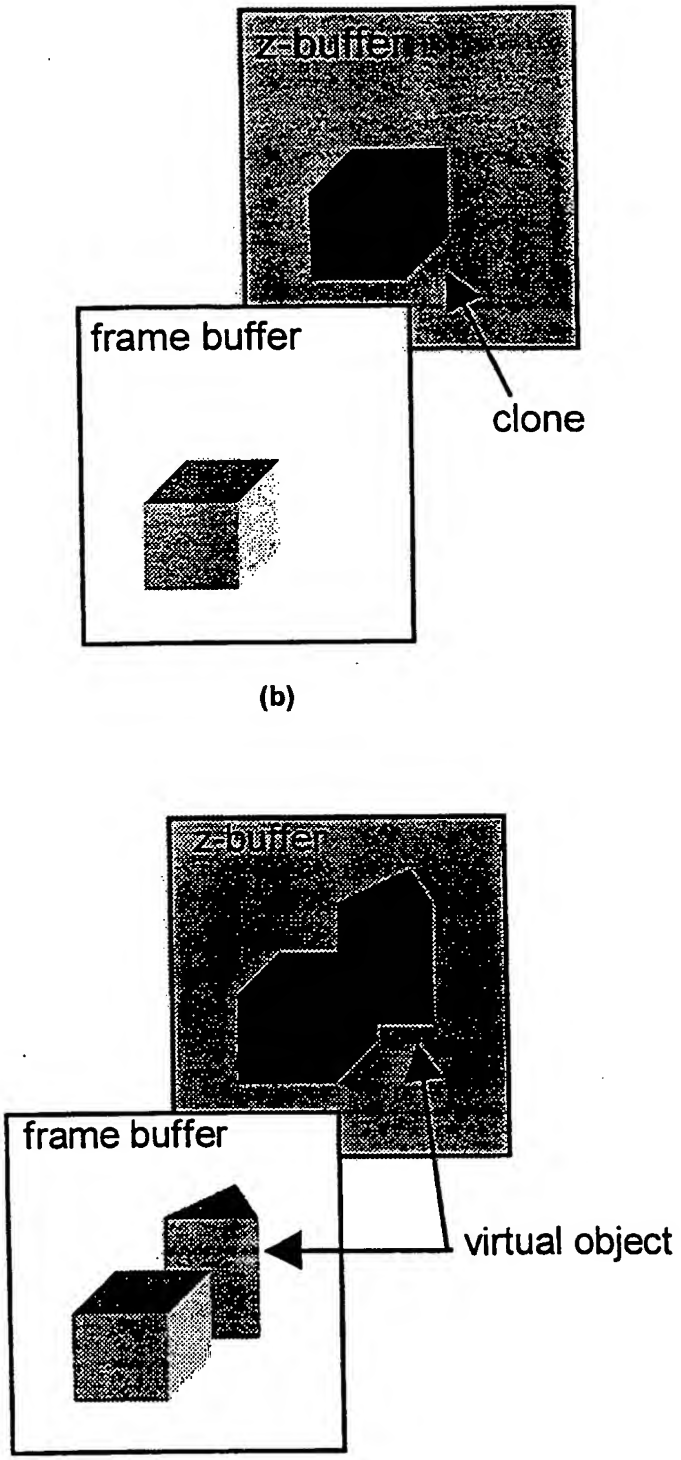


FIG. 10 (a)



(a)

FIG. 10 (b) & (c)


z-buffer
frame buffer
clone
z-buffer
frame buffer
virtual object

(b)

(c)

FIG. 11

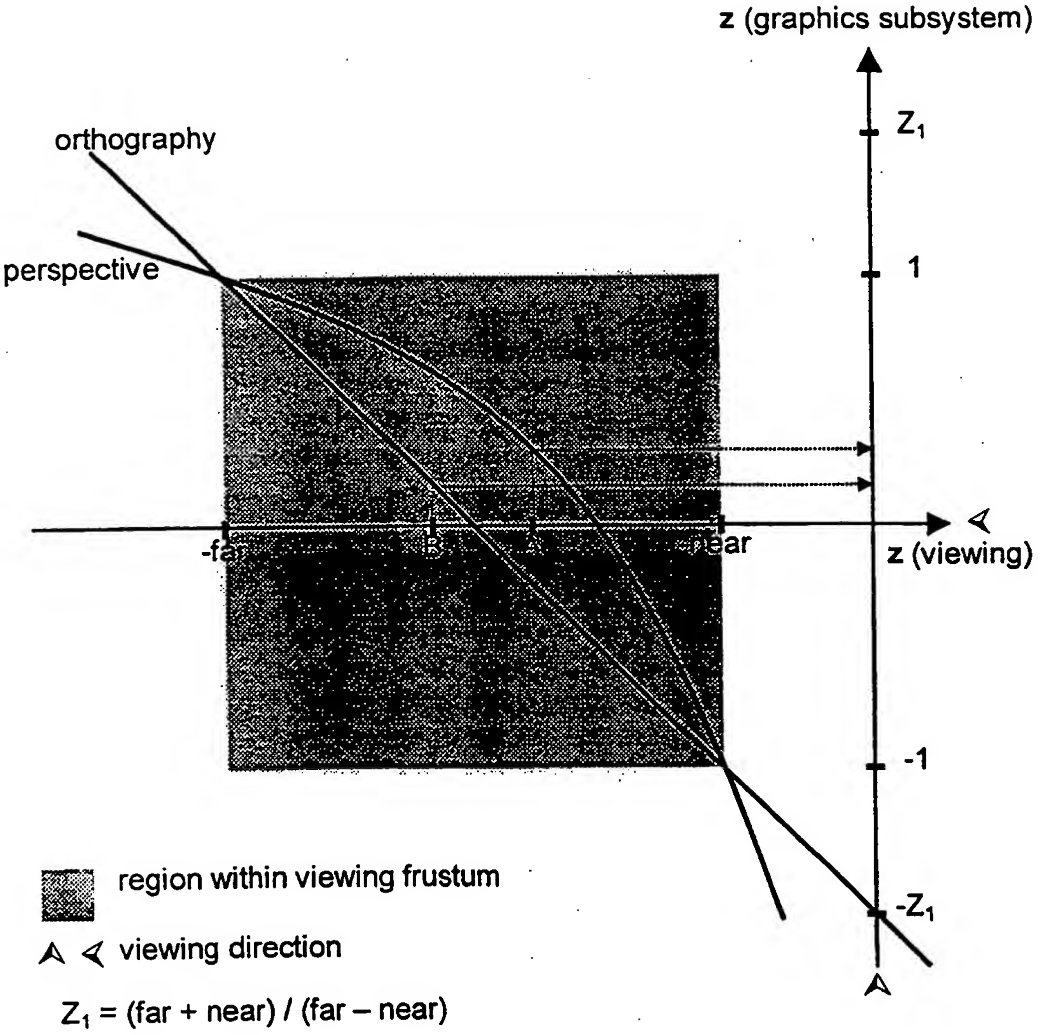
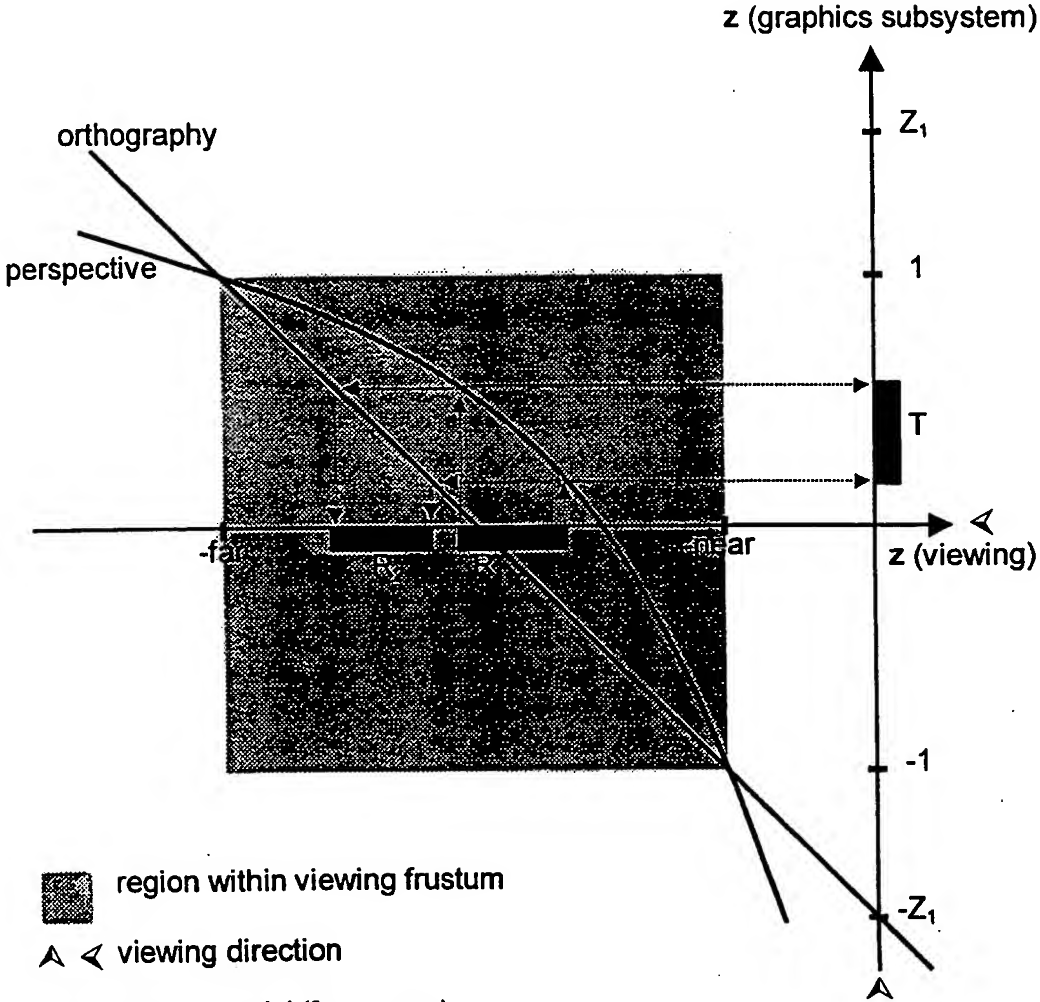

z (graphics subsystem)
Z_1
1
-1
$-Z_1$
orthography
perspective
-far
B
A
near
z (viewing)
▨ region within viewing frustum
⮝ ⮜ viewing direction
Z_1 = (far + near) / (far – near)

FIG. 12



Z_1 = (far + near) / (far – near)